UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/523,877	03/13/2000	Peter Warnes	ARC.005A	6131

27299 7590 04/27/2004

GAZDZINSKI & ASSOCIATES
11440 WEST BERNARDO COURT, SUITE 375
SAN DIEGO, CA 92127

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	22

DATE MAILED: 04/27/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/523,877	WARNES ET AL.
	Examiner	Art Unit
	David J. Huisman	2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 March 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,14-20,25-32 and 37-52* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5,14-20,25-32 and 37-52* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *13 March 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-5, 14-20, 25-32, and 37-52 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: #21. Amendment "F" as received on 3/4/2004.

Claim Objections

3. Claim 14 is objected to because of the following informalities: The applicant states "an instruction set comprising a plurality of instruction words" multiple times (lines 3 and 5 of claim 14). The applicant should eliminate one of these occurrences. Appropriate correction is required.

4. Claim 20 is objected to because of the following informalities: In line 9, please delete "(iv)" and the semicolon at the end of that line. Appropriate correction is required.

5. Claim 37 is objected to because of the following informalities: Please insert a period at the end of the claim. Appropriate correction is required.

6. Claim 40 is objected to because of the following informalities: Applicant claims at least four discrete modes. However, in the last line applicant claims that each mode is unique with respect to the other three modes; that is, there are exactly four modes. Applicant should replace "at lest four discrete modes" with --four discrete modes--. Appropriate correction is required.

7. Claim 49 is objected to because of the following informalities: The applicant states "an instruction set comprising a plurality of instruction words" multiple times (lines 3 and 4 of claim

49). The applicant should eliminate one of these occurrences. Appropriate correction is required.

8. Claim 50 recites the limitations "said data storage device" in line 3, and the limitation "said at least two data bits" in the second to last line. There is insufficient antecedent basis for these limitations in the claim.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-5, 14-18, 20, 25, 29, 44-49, and 50-52 are rejected under 35 U.S.C. 102(b) as being anticipated by Lee et al., U.S. Patent No. 4,755,966 (as applied in the previous Office Action and herein referred to as Lee).

11. Referring to claim 1, Lee has taught a method of controlling the execution of instructions within a pipelined processor, comprising:

a) providing an instruction set comprising a plurality of instruction words. Lee discloses in column 3, lines 46-47, that each instruction within the instruction set contains a 6-bit opcode, which means a total of 64 instructions could exist within the system. Lee also discloses in column 6, lines 36-39, that the system contains a floating-point unit, which means floating-point instructions would exist.

b) each of said instruction words comprising a plurality of data bits. See column 3, lines 40-42. Each instruction is 32 bits.

c) at least one of said words comprising a jump instruction having at least one user-configurable mode and at least one user-definable mode associated therewith, said user-configurable and user-definable modes each being specified by the same ones of said plurality of data bits, said at least one user-definable mode not being predetermined in terms of function. See Fig.2, component 102, and note the use of a branch (jump) instruction. Also, it should be realized that that the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). This is equivalent to at least one user-configurable mode. In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) assigning one of a plurality of values to said ones of said data bits of said at least one jump instruction. See column 3, lines 46-51. Lee discloses that each branch instruction contains a nullify bit and a displacement sign bit. These bits can be assigned a value (0 or 1) as shown in Fig.3.

e) controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said ones of data bits when said at least one jump instruction is decoded. See column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

12. Referring to claim 2, Lee has taught a method as described in claim 1. Lee has further taught that the act of assigning comprises identifying said ones of data bits within said at least one jump instruction and assigning one of two discrete values to each of said ones of data bits, the combination of said two discrete values representing at least three jump delay slot modes within said processor. See column 5, lines 7-46 for a description of the 5 different jump delay slot modes shown in Fig.2 In essence, the nullify bit (Fig.1, component 507) and the displacement sign bit (Fig.1, component 508) are checked by the system and the combination of those values will determine the jump delay slot mode. It is inherent that each bit would be assigned one of two discrete values (i.e. 0 or 1) since a processor only understands binary values.

13. Referring to claim 3, Lee has taught a method as described in claim 2. Lee has further taught that the act of controlling the execution based on said discrete values comprises selecting at least one mode from the group comprising:

a) executing said at least one subsequent instruction under all circumstances. See column 5, lines 50-53, and Fig.3.

b) executing said at least one subsequent instruction only if a jump occurs. See column 5, lines 53-57, and Fig.3. The delay slot instruction will be executed when a jump occurs and the displacement is negative. The delay slot instruction will not be executed if a jump does not occur and the displacement is negative.

c) stalling the pipeline or inserting a bubble into the pipeline if a jump occurs. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore, in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle.

14. Referring to claim 4, Lee has taught a method as described in claim 3. Lee has further taught that at least one jump instruction comprises a conditional branch instruction. See column 2, lines 62-64, and note the conditional component 202.

15. Referring to claim 5, Lee has taught a method as described in claim 1. Furthermore, note that the displacement sign bit is assigned one value as is the nullify bit. Therefore, claim 5 is rejected for the same reasons set forth in the rejection of claim 3 above.

16. Referring to claim 14, Lee has taught a digital processor comprising:

a) a processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words. Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39. Furthermore, it is inherent that the processor will decode and execute multiple instructions (as established in the rejection of claim 1) from an instruction set.

b) a data interface between said processor core and an information storage device. It is inherent that in order for a processor to execute instructions, they must be stored in the processor's memory. Therefore, the instructions would be stored in an information storage device that is directly accessible by the processor.

c) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump

delay slot modes and at least one user-defined mode, said jump delay slot modes and at least one user-defined mode each being specified by the same portions of said data, said at least one user-defined mode not being predetermined in terms of function, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are user-configurable in that a number of different modes which are achieved in part by the user. For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes or not is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

17. Referring to claim 15, Lee has taught a digital processor as described in claim 14. Furthermore, claim 15 is rejected for the same reasons set forth in the rejection of claim 3 above.

18. Referring to claim 16, Lee has taught a digital processor as described in claim 14. Lee has further taught that at least one jump instruction comprises a conditional branch instruction having an associated logical condition, the execution of a jump to the address within said

information storage device specified by said at least one conditional branch instruction being determined by said logical condition. See Fig.3, component 202, and note that the branch will be taken or not taken based on some condition. Furthermore, it is inherent that the execution of the branch will cause a jump to the address (specified by the instruction) within the information storage device.

19. Referring to claim 17, Lee has taught a digital processor having at least one pipeline and an associated data storage device, wherein the execution of instructions within said at least one pipeline is controlled by the method comprising:

a) storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a user-configurable branch instruction having a plurality of unique functional modes exclusively associated with respective ones of unique combinations of a plurality of mode control bits, said branch instruction directing branching to a first address within said data storage device. As discussed above in the rejection of claim 14, Lee has taught an instruction set with a plurality of instructions that would inherently comprise a plurality of bits and would inherently be stored in a data storage device in order to processor-accessible and executable. Furthermore, it is inherent that a branch will cause a jump to a specified address within the data storage device. Finally, it should be noted that these branch instructions are user-configurable instructions having unique functional modes associated with unique mode control bit combinations. For instance, one basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)

01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a unique delay slot mode; that is, different functionality is achieved with each combination of bits.

b) assigning one of a plurality of values to each of said mode control bits of said at least one branch instruction. See column 3, lines 46-51. Lee discloses that each instruction contains 32 data bits of information. This information includes source registers, displacements, an opcode, condition fields, and a nullify bit and a displacement sign bit (mode control bits). Each of these bits is assigned a value (0 or 1). For instance, if the user decides to use a branch instruction, then he/she must set the nullify bit to the desired state and the displacement bit will be determined based on whether the user specifies the branch to be a forward or backwards branch.

c) decoding said at least one branch instruction including said one values. Since, the values of the nullify bit and displacement sign bit are part of each branch instruction, it follows that the value will be decoded as the branch instruction is decoded. See Fig.1 and column 3, lines 46-51.

d) determining whether to execute an instruction within said pipeline in a stage preceding that of said at least one branch instruction based at least in part on said assigned values. The delay slot instruction would be in a pipeline stage preceding that of the branch instruction. And, Lee discloses a system in which the execution of the delay slot instruction is determined based on the value of the nullify bit and/or displacement bit.

e) branching to said first address based on said at least one branching instruction. Recall from claim 16, that it is the inherent nature of a branch instruction (when taken) to jump to a specified address.

f) performing, based at least in part on said act of decoding said assigned values, at least one other function dictated by the unique functional mode associated with said assigned values. Looking at Fig.2, component 113, and Fig.3, it should be realized that depending on the mode of the branch instruction, the delay slot instruction will either be executed or not executed. If it is determined that the delay slot instruction will be executed, then it will eventually be performed following the branch. If the delay slot instruction is an "ADD" instruction, for instance, an addition function will be performed.

20. Referring to claim 18, Lee has taught a processor as described in claim 17. Furthermore, it is inherent that the data bits comprise binary data. A processor can only recognize and understand zeroes and ones.

21. Referring to claim 20, Lee has taught a method of controlling program operation of a multi-stage pipelined digital processor having a data storage device in data communication therewith, comprising:

a) storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a branch instruction directing branching to a first address within said data storage device based on a first parameter. This portion of claim 20 is rejected for the same reasons set forth in the rejection of claim 17 above. Furthermore, the first parameter could be considered the logic that determines whether the branch is taken or not taken (ex. a flag is checked and compared to a value).

b) defining a plurality of jump delay slot modes comprising:

(i) executing a subsequent instruction under all circumstances. See column 5, lines 50-53, and Fig.3.

(ii) executing a subsequent instruction only if jumping occurs. See column 5, lines 53-57, and Fig.3. The delay slot instruction will be executed when a jump occurs and the displacement is negative. The delay slot instruction will not be executed if a jump does not occur and the displacement is negative.

(iii) stalling the pipeline for one cycle if jumping occurs. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore, in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle.

c) assigning said plurality of jump modes to at least two of said data bits of said at least one branch instruction, wherein changing of any one or more of said at least two data bits will always specify a different functional mode. Note that both the displacement bit and nullify bit control the jump mode. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that by changing one or more (at least one) bit, a different functional mode will always be realized. For instance, taking 10 and 11 as an example, by changing the second bit (sign bit), a different mode will be realized. That is you can go from being in either

DBE/BND mode to being in DBN/BED mode, and vice versa. As another example, if you change the nullify bit of 00, you end up with 10, so you go from FDS mode to DBE/BND mode.

d) decoding said at least one branch instruction including said at least two data bits. Note that when the branch instruction is decoded, the nullify and displacement bits are decoded as well in order to determine the mode. See Fig.3.

e) controlling said pipeline based at least in part on said at least two data bits and said first parameter. Again, the first parameter would involve the condition field of the branch instruction. See column 3, line 49. For instance, this parameter might specify to perform a jump if a certain value is greater-than or equal to 0. In addition, the nullify and displacement bits play a role in controlling the operation of the pipeline. See Fig.3.

22. Referring to claims 25 and 29, Lee has taught a method and digital processor as described in claims 1 and 14, respectively. Lee has further taught that at least one of said plurality of instruction words comprises an op-code and a plurality of fields, each of said fields comprising a plurality of bits (see column 3, lines 46-51), said at least one instruction word being encoded according to the method comprising:

a) associating a first of said fields with a first data source. See column 3, lines 47-48 (component 503).

b) associating a second of said fields with a second data source. See column 3, lines 48-49 (component 504).

c) performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code. See column 3, lines 51-55. In this case, the

opcode specifies a compare and branch instruction where the comparison is performed between the contents of the two specified registers.

23. Referring to claim 44, Lee has taught a method as described in claim 1. Lee has further taught that said at least one user-definable mode comprises a non-jump or branch related mode. Clearly from Fig.3, Lee has taught branch related modes, as each mode is involved with a branch instruction. It should be realized that applicant's use of the word "or" allows Lee to anticipate the claim in this situation.

24. Referring to claim 45, Lee has taught a method as described in claim 1. Lee has further taught that the operation of said at least one user-configurable or user-definable modes is not determined by a displacement value. See Fig.3 and note that when the nullify bit is off, the displacement value has no part in determining that the delay slot instruction is executed.

25. Referring to claim 46, Lee has taught a method as described in claim 1. Lee has further taught that the operation of said at least one user-configurable or user-definable modes is not dependent on an address calculation value. See Fig.3 and note that when the nullify bit is off, an address calculation value has no part in determining that the delay slot instruction is executed.

26. Referring to claim 47, Lee has taught a method as described in claim 44. Lee has further taught that the operation of said at least one user-configurable or user-definable modes is not determined by a displacement value. See Fig.3 and note that when the nullify bit is off, the displacement value has no part in determining that the delay slot instruction is executed.

27. Referring to claim 48, Lee has taught a method as described in claim 44. Lee has further taught that the operation of said at least one user-configurable or user-definable modes is not

dependent on an address calculation value. See Fig.3 and note that when the nullify bit is off, an address calculation value has no part in determining that the delay slot instruction is executed.

28. Referring to claim 49, Lee has taught a digital processor comprising:

a) a processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words. Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39. Furthermore, it is inherent that the processor will decode and execute multiple instructions from an instruction set (as established in the rejection of claim 1).

b) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump delay slot modes, said jump delay slot modes each being specified by the same portions of said data without reference to an address calculation metric, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are user-configurable in that a number of different modes which are achieved in part by the user. For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). Also, a mode is specified without reference to an address calculation metric in that the entire displacement taught by Lee is an address calculation metric because the entire displacement is required to perform an address calculation. In selecting a mode, the entire displacement is not referenced. Instead, just

the sign bit of the displacement is used and this bit in itself does not constitute an address calculation metric.

29. Referring to claim 50, Lee has taught a method of controlling program operation of a multi-stage pipelined digital processor, comprising:

a) storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a branch instruction directing branching to a first address within said data storage device based on a first parameter. This portion of claim 20 is rejected for the same reasons set forth in the rejection of claim 17 above. Furthermore, the first parameter could be considered the logic that determines whether the branch is taken or not taken (ex. a flag is checked and compared to a value).

b) defining a plurality of constrained jump delay slot functional modes comprising:

(i) executing a subsequent instruction under all circumstances. See column 5, lines 50-53, and Fig.3.

(ii) executing a subsequent instruction only if jumping occurs. See column 5, lines 53-57, and Fig.3. The delay slot instruction will be executed when a jump occurs and the displacement is negative. The delay slot instruction will not be executed if a jump does not occur and the displacement is negative.

(iii) stalling the pipeline for one cycle if jumping occurs. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore,

in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle.

c) assigning said plurality of jump modes to a plurality of said data bits of said at least one branch instruction, each of said functional modes being constrained to only one of a plurality of unique combinations of said plurality of bits. See Fig.3 and note that both the displacement bit and nullify bit control the jump modes. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

d) decoding said at least one branch instruction including said plurality of data bits. Note that when the branch instruction is decoded, the nullify and displacement bits are decoded as well in order to determine the mode. See Fig.3.

e) controlling said pipeline based at least in part on said at least two data bits and said first parameter. Again, the first parameter would involve the condition field of the branch instruction. See column 3, line 49. For instance, this parameter might specify to perform a jump if a certain value is greater-than or equal to 0. In addition, the nullify and displacement bits play a role in controlling the operation of the pipeline. See Fig.3.

30. Referring to claim 51, Lee has taught a digital processor having at least one pipeline (Fig.5) and adapted to interface with an associated data storage device and having an instruction

disposed at least partly within said data storage device (it is inherent that instructions must be stored in some storage device), said instruction set comprising a plurality of instruction words, at least one of said instruction words comprising a branch instruction having a plurality of functional modes associated with respective ones of combinations of a plurality of mode control bits (see Fig.3 and note modes depend on nullify and displacement bits), said branch instruction directing branching to a first address within said data storage device (this is the inherent operation of a branch instruction), wherein the execution of instructions within said at least one pipeline is controlled by the method comprising:

a) reading said mode control bits of said at least one branch instruction. See Fig.3, and note that the mode bits must be read in order to determine what happens with the delay slot instruction.

b) determining whether to execute an instruction within said pipeline in a stage different than that of said at least one branch instruction based at least in part on said read mode control bits and irrespective of the direction of said branching. It is inherent that the delay slot instruction would be in a pipeline stage different than that of the branch instruction (since it is executed after the branch instruction). And, from Fig.3, it should be realized that if the nullify bit is off, then the delay slot instruction will be executed regardless of the branching direction.

e) branching to said first address based on said at least one branching instruction. Again, it is the inherent nature of a branch instruction (when taken) to jump to a specified address.

f) performing at least one other function dictated by the functional mode associated with said read bits. Looking at Fig.2, component 113, and Fig.3, it should be realized that depending on the mode of the branch instruction, the delay slot instruction will either be executed or not executed. If it is determined that the delay slot instruction will be executed, then it will

eventually be performed following the branch. If the delay slot instruction is an "ADD" instruction, for instance, an addition function will be performed.

31. Referring to claim 52, Lee has taught a digital processor having at least one pipeline (Fig.5) and adapted to interface with an associated data storage device and having an instruction disposed at least partly within said data storage device (it is inherent that instructions must be stored in some storage device), said instruction set comprising a plurality of instruction words, at least one of said instruction words comprising a branch instruction having a plurality of functional modes associated with respective ones of a plurality of combinations of a plurality of mode control bits (see Fig.3 and note modes depend on nullify and displacement bits), at least one of said plurality of combination and the logical functions associated therewith being adapted for specification by a user (it should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.), said branch instruction directing branching to a first address within said data storage device (this is the inherent operation of a branch instruction), wherein the execution of instructions within said at least one pipeline is controlled by said mode control bits of said at least one branch instruction (see Fig.3 and note that the delay slot instruction's execution is dependent on the mode control bits of the branch instruction).

Claim Rejections - 35 USC § 103

32. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

33. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied above, in view of Hennessy and Patterson, Computer Architecture - A Quantitative Approach, 2nd Edition, 1996 (herein referred to as Hennessy).

34. Referring to claim 19, Lee has taught a processor as described in claim 17. Lee has further taught a 4-stage pipeline with an instruction address generation stage, an instruction fetch stage, an execute stage, and a write stage. See Fig.5 and column 6, lines 56-64. Lee has not explicitly taught a stage just for decoding. However, Lee has taught that decoding is done in one of the aforementioned stages. See column 7, lines 12-16. Lee further states that the execution of instructions can be pipelined to any depth desired. See column 6, lines 63-64. Official Notice is taken that decode pipeline stages and their advantages are well known and expected in the art. The actual implementation of the pipeline is a designer's preference but Lee has taught a system in which any size pipeline would suffice. In addition, Hennessy has shown that as the number of pipeline stages increases, then the average time per instruction decreases. See page 126 and note the formula (the more stages, the lower the amount of time per instruction). As a result, if desired, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a pipeline with a separate decode stage (thereby increasing the number of stages in Lee by 1), so that the amount of time required to execute an instruction would decrease.

35. Claims 26-28 and 30-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied to claims 1 and 14, respectively, in view of Kawasaki et al., U.S. Patent No. 5,530,965 (herein referred to as Kawasaki).

36. Referring to claims 26, 27, 30, and 31, Lee has taught a method and digital processor as described in claims 1, 25, 14, and 29, respectively.

a) Lee has further taught of providing an instruction word having an opcode and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits. Note from column 3, lines 46-51, that the instruction format includes an 11 bit displacement field, which holds an immediate constant for branching purposes.

b) Lee has not explicitly taught selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit

positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

37. Referring to claims 28 and 32, Lee in view of Kawasaki has taught a method and digital processor as described in claims 27 and 31, respectively. Recall that Lee has taught an

instruction format that includes a 6-bit opcode field, meaning Lee's system has the ability of choosing between 64 different instructions if necessary. With the exception of branching, Lee has not explicitly stated any other instructions that have been implemented. However, it is inherent that other instructions would exist so that the processor can perform useful operations. A processor that does nothing but branching would do nothing useful. Furthermore, in column 3, lines 46-51, Lee has disclosed that his system is a register-register (load-store) architecture, i.e. where main memory is only accessed through load and store operations and operations are performed on values in registers. This is known because Lee has implemented an instruction format with two register operands. A move (mov) instruction is also well known in the art and is explicitly shown in Kawasaki. More specifically, Kawasaki has shown multiple versions of a move instruction. One version moves a short immediate value into a register (as shown in column 51, lines 51 and 55-56) and another version includes moving a value from one register to another (as shown in column 52, line 53). A person of ordinary skill in the art would expect to find these common move instructions within the system of Lee because they allow a programmer to move an initial value into a register as well as temporarily store a register value into another register. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have at least one instruction word having a plurality of fields and said at least one instruction word having a short immediate value comprise the same instruction word(s); in this case, the move instruction, taught by Kawasaki.

38. Claims 37-40 and 42-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied above, in view of Wirthlin et al., The Nano Processor: a Low Resource Reconfigurable Processor, 1994 (herein referred to as Wirthlin).

39. Referring to claim 37, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having a plurality of user-configurable modes determined by a plurality of bits controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline, each of said modes being constrained to only one of a plurality of unique combinations of said plurality of bits.. See Fig.3 and note the configurable modes, which are specified by the nullify and displacement bits (Fig.2, field 507). By setting or resetting these bits, the user will configure the system to either never nullify a delay slot instruction or sometimes nullify a delay slot instruction. Also, as shown in the table in the rejection of claim 35(b) above, there are a plurality of user-configurable modes for each branch instruction determined by a plurality of bits. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

b) Lee has not taught that the instruction set comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and a custom instruction set (extension instruction) is used. See page 25, sections 3.1.1

and 3.1.2. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task.

40. Referring to claim 38, Lee has taught a pipelined digital processor comprising:

a) a branch instruction including two data bits defining four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline. See the nullify and displacement bits from Fig.3. These two bits define at least four delay sot modes as shown in the rejection of claim 35(b) above. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not taught that the instruction set comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and a custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set

and at least one extension instruction. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task.

c) said execution is controlled without regard to a branch direction metric. The examiner agrees that when the nullify bit is on, the branch direction is taken into consideration. However, when the nullify bit is off, execution is controlled without regard to the branch direction. Therefore, the claim is still anticipated.

41. Referring to claim 39, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having at least one mode controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline using a plurality of data bits, at least one of the particular combinations of data bits and the logical functions associated therewith being adapted for assignment by a user. As shown in Fig.3, there are a plurality of modes in which each branch can operate. The mode is dependent on the nullify bit and the displacement bit. Also, it should be realized that the user that is writing the program will be setting the nullify bit as well as the displacement bit by choosing whether the branch will be a forward or backwards branch. Therefore, they, along with the logical functions associated with the mode bits, are adapted for assignment by a user. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed it taught by Lee, as each branch instruction will have a combination assigned to it.

b) Lee has not taught that the instruction set comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and a custom instruction set (extension instruction) is used. See page 25, sections 3.1.1

and 3.1.2. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task.

42. Referring to claim 40, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having at least four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline. See the nullify and displacement bits from Fig.3. These two bits define at least four delay sot modes as shown in the rejection of claim 35(b) above. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not taught that the instruction set comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and a custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set

and at least one extension instruction. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task.

c) each of said modes provides unique functionality with respect to the other three modes. See Fig.3. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a mode with unique functionality with respect to the other three modes.

43. Referring to claim 42, Lee has taught a digital processor as described in claim 40. Lee has further taught that first and second of said at least four modes implement one- and two-cycle stalls within said pipeline, respectively. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore, in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle. Also, see Fig.5, column 5, lines 58-63, and column 7, lines 21-25. Note that pipeline includes four stages: an address generation stage (A), a fetch stage (F), an execution stage (E), and a write stage (W). It has been disclosed that the target address of the branch is not determined until the end of the execution stage. By this time, the delay slot instruction will have been fetched, and the predicted target address will be supplied to the program counter for fetching in the next cycle. See column 7, lines 6-10. If the delay slot instruction ends up being nullified (based on the displacement and

nullify bits) and the predicted target is incorrect, then both instructions will need to be cancelled, resulting in a 2-cycle stall.

44. Referring to claim 43, Lee has taught a digital processor as described in claim 40. Lee has further taught that at least one of said at least four modes operates without respect to a branch displacement metric. See Fig.3 and note that when the nullify bit is off, the displacement value has no part in determining that the delay slot instruction is executed.

Response to Arguments

45. Applicant's arguments filed on January 9, 2004, have been fully considered but they are not persuasive. In general, the responses to the arguments below mirror those set forth in the rejections above.

46. In the remarks, Applicant argues the novelty/rejection of claims 1 and 14 on page 13 of the remarks, in substance that:

> ...in Lee, "the programmer has no choice or capability to define or specify underlying logical relationships associated with the user-configurable bits that are set."

47. These arguments are not found persuasive for the following reasons:

a) With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running.

48. In the remarks, Applicant argues the novelty/rejection of claim 17 on page 14 of the remarks, in substance that:

"Lee in no way teaches or suggests providing unique functional modes exclusively associated with respective ones of unique combinations of a plurality of mode control bits."

49. These arguments are not found persuasive for the following reasons:

a) One basic breakdown of the delay slot modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a unique delay slot mode; that is, different functionality is achieved with each combination of bits.

50. In the remarks, Applicant argues the novelty/rejection of claim 20 on page 15 of the remarks, in substance that:

Lee does not teach "that changing of any one or more of the data bits will always specify a different functional mode."

51. These arguments are not found persuasive for the following reasons:

a) Note that both the displacement bit and nullify bit control the jump mode. One basic breakdown of the modes in Lee is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that by changing one or more (at least one) bit, a different functional mode will always be realized. For instance, taking 10 and 11 as an example, by changing the

second bit (sign bit), a different mode will be realized. That is, you can go from being in either DBE/BND mode to being in DBN/BED mode, and vice versa. As another example, if you change the nullify bit of 00, you end up with 10, so you go from FDS mode to DBE/BND mode.

52. In the remarks, Applicant argues the novelty/rejection of claims 37 and 40 on page 16 of the remarks, in substance that:

> Lee does not teach "each of the designated modes being constrained to only one of a plurality of unique combinations of the mode bits."

53. These arguments are not found persuasive for the following reasons:

a) One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

54. In the remarks, Applicant argues the novelty/rejection of claim 38 on page 16 of the remarks, in substance that:

> "Lee must necessarily consider the branch direction bit." Therefore, Lee does not teach "execution being controlled without regard to a branch direction metric."

55. These arguments are not found persuasive for the following reasons:

a) The examiner agrees that when the nullify bit is on, the branch direction is taken into consideration. However, when the nullify bit is off, execution is controlled without regard to the branch direction. Therefore, in this situation, Lee still anticipates applicant's claim.

56. In the remarks, Applicant argues the novelty/rejection of claim 39 on pages 16-17 of the remarks, in substance that:

> Lee does not teach "limitations relating to the particular combinations of data bits and the logical functions associated therewith being adapted for assignment by a user."

57. These arguments are not found persuasive for the following reasons:

a) It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality (logical function) is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

Conclusion

58. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
April 26, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100